J 5

```
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUU         UUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUU         UUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUUUUUUUUUUUUUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUUUUUUUUUUUUUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUUUUUUUUUUUUUU
```

FILEID**OBJECT   E 5

```
  OOOOOO    BBBBBBBB            JJ  EEEEEEEEEE    CCCCCCCC  TTTTTTTTTT
  OOOOOO    BBBBBBBB            JJ  EEEEEEEEEE    CCCCCCCC  TTTTTTTTTT
OO       OO BB      BB          JJ  EE          CC              TT
OO       OO BB      BB          JJ  EE          CC              TT
OO       OO BB      BB          JJ  EE          CC              TT
OO       OO BB      BB          JJ  EE          CC              TT
OO       OO BBBBBBBB            JJ  EEEEEEEE    CC              TT
OO       OO BBBBBBBB            JJ  EEEEEEEE    CC              TT
OO       OO BB      BB  JJ      JJ  EE          CC              TT
OO       OO BB      BB  JJ      JJ  EE          CC              TT
OO       OO BB      BB  JJ      JJ  EE          CC              TT         ....
OO       OO BB      BB  JJ      JJ  EE          CC              TT         ....
  OOOOOO    BBBBBBBB      JJJJJJ    EEEEEEEEEE    CCCCCCCC      TT         ....
  OOOOOO    BBBBBBBB      JJJJJJ    EEEEEEEEEE    CCCCCCCC      TT         ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0001  0 MODULE object            (IDENT='V04-000',
0002  0                          ADDRESSING_MODE(EXTERNAL=GENERAL))
0003  1 = BEGIN
0004  1
0005  1 !*************************************************************************
0006  1 !*                                                                       *
0007  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
0008  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.               *
0009  1 !*  ALL RIGHTS RESERVED.                                                 *
0010  1 !*                                                                       *
0011  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
0012  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
0013  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
0014  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
0015  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
0016  1 !*  TRANSFERRED.                                                         *
0017  1 !*                                                                       *
0018  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
0019  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0020  1 !*  CORPORATION.                                                         *
0021  1 !*                                                                       *
0022  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
0023  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.              *
0024  1 !*                                                                       *
0025  1 !*                                                                       *
0026  1 !*************************************************************************
0027  1
0028  1 !++
0029  1 ! Facility:     Command Definition Utility, Object File Module
0030  1 !
0031  1 ! Abstract:     This module contains the routines necessary to create a
0032  1 !               object file from a set of CLDs.  Once the CLDs are compiled,
0033  1 !               the resulting tables are transformed into an object records
0034  1 !               and placed in a file.
0035  1 !
0036  1 ! Environment:  Standard CDU environment.
0037  1 !
0038  1 ! Author:       Paul C. Anagnostopoulos
0039  1 ! Creation:     24 January 1983
0040  1 !
0041  1 ! Modifications:
0042  1 !
0043  1 !       V04-001 KPL0001         Peter Lieberwirth       28-Jun-1984
0044  1 !               Record Attributes of object module should be NULL, not
0045  1 !               CR, for consistency with all other object modules.
0046  1 !
0047  1 !--
0048  1
0049  1
0050  1 library 'sys$library:lib';
0051  1 require 'clitabdef';
0376  1 require 'cdureq';
```

```
;     1 !       T A B L E   O F   C O N T E N T S
;     1 !       ---------   ---   -------------
;     1
;     1 forward routine
;     1         cdu$prepare_object_file: novalue,
;     1         cdu$write_object_file: novalue,
;     1         write_header_records: novalue,
;     1         write_global_symbol_record: novalue,
;     1         write_psect_record: novalue,
;     1         write_table_records: novalue,
;     1         write_user_routine_records: novalue,
;     1         write_eom_record: novalue;
;     1
;     1
;     1 !       E X T E R N A L   R E F E R E N C E S
;     1 !       ---------------   -------------------
;     1
;     1 external routine
;     1         cdu$collect_table_blocks,
;     1         cdu$lookup_child,
;     1         cdu$report_rms_error,
;     1         cli$get_value,
;     1         lib$free_vm,
;     1         lib$get_vm;
;     1
;     1 external
;     1         cdu$facility_string: descriptor,
;     1         cdu$gl_root_node: ref node,
;     1         cdu$gl_table: pointer;
;     1
;   P 1 $shr_msgdef(cdu,17,local,
;   P 1         (openout,severe),
;   P 1         (writeerr,severe)
;     1         );
```

```
!       O B J E C T   F I L E   C O N T R O L   B L O C K S
!       -----------   -------   -------------   -----------

! The following items define the RMS control blocks needed to create and
! write the object file.

own
        object_related_rsa: block[nam$c_maxrss,byte],
        object_related_nam: $nam(),

        object_esa: block[nam$c_maxrss,byte],
        object_rsa: block[nam$c_maxrss,byte],
        object_nam: $nam(
                        esa=object_esa,
                        ess=%allocation(object_esa),
                        rlf=object_related_nam,
                        rsa=object_rsa,
                        rss=%allocation(object_rsa)
                        ),

        dbuffer(object_spec,nam$c_maxrss),
        object_fab: $fab(
                        dnm='.OBJ',
                        fna=object_spec+8,
                        fns=%allocation(object_spec)-8,
                        fac=put,
                        fop=<sqo,nam,ofp>,
                        nam=object_nam,
                        org=seq,
                        rfm=var
                        ),

        object_rab: $rab(
                        fab=object_fab,
                        rac=seq,
                        rop=wbh
                        );
```

```
;  127          0861    1 !++
;  128          0862    1 ! Description:  This routine is called to prepare the object file for
;  129          0863    1 !               writing of the object records.  All we do is save enough
;  130          0864    1 !               information so that we can create it after the CLDs are
;  131          0865    1 !               compiled.
;  132          0866    1 !
;  133          0867    1 ! Parameters:   cld_fab         By reference, the FAB used to read the first
;  134          0868    1 !                               CLD file.
;  135          0869    1 !
;  136          0870    1 ! Returns:      Nothing.
;  137          0871    1 !
;  138          0872    1 ! Notes:
;  139          0873    1 !--
;  140          0874    1
;  141          0875    1 GLOBAL ROUTINE cdu$prepare_object_file(cld_fab: pointer)          : novalue
;  142          0876    2 = BEGIN
;  143          0877    2
;  144          0878    2 bind
;  145          0879    2         cld_nam = .cld_fab[fab$l_nam]: block[,byte];
;  146          0880    2
;  147          0881    2
;  148          0882    2 ! We don't want to create the object file now, because the CLDs may have
;  149          0883    2 ! errors and we'll end up with a null file.  However, we do want to save
;  150          0884    2 ! the NAM block and resultant strings from the CLDs so we can used them as
;  151          0885    2 ! the related name when we create the object file.
;  152          0886    2
;  153          0887    2 ch$move(.cld_nam[nam$b_bln],cld_nam, object_related_nam);
;  154          0888    2 ch$move(.cld_nam[nam$b_rss],.cld_nam[nam$l_rsa], object_related_rsa);
;  155          0889    2
;  156          0890    2 return;
;  157          0891    2
;  158          0892    1 END;


                                                        .TITLE  OBJECT
                                                        .IDENT  \V04-000\

                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                              4A  42  4F  2E  00000 P.AAA:  .ASCII  \.OBJ\                                  ;

                                                        .PSECT  $OWN$,NOEXE,2

                                           00000 OBJECT_RELATED_RSA:
                                                        .BLKB   255
                                           000FF        .BLKB   1
                                       02  00100 OBJECT_RELATED_NAM:
                                                        .BYTE   2                                       ;
                                       60  00101        .BYTE   96                                      ;
                                       00  00102        .BYTE   0                                       ;
                                       00  00103        .BYTE   0                                       ;
                                 00000000  00104        .LONG   0                                       ;
                                       00  00108        .BYTE   0                                       ;
                                       00  00109        .BYTE   0                                       ;
                                       00  0010A        .BYTE   0                                       ;
                                       00  0010B        .BYTE   0                                       ;
                                 00000000  0010C        .LONG   0                                       ;
```

```
00000000  00110              .LONG    0                                          ;
    0000# 00114              .WORD    0[8]                                       ;
    0000# 00124              .WORD    0[3]                                       ;
    0000# 0012A              .WORD    0[3]                                       ;
00000000  00130              .LONG    0                                          ;
00000000  00134              .LONG    0                                          ;
      00  00138              .BYTE    0                                          ;
      00  00139              .BYTE    0                                          ;
      00  0013A              .BYTE    0                                          ;
      00  0013B              .BYTE    0                                          ;
      00  0013C              .BYTE    0                                          ;
      00  0013D              .BYTE    0                                          ;
      00# 0013E              .BYTE    0[2]                                       ;
00000000  00140              .LONG    0                                          ;
00000000  00144              .LONG    0                                          ;
00000000  00148              .LONG    0                                          ;
00000000  0014C              .LONG    0                                          ;
00000000  00150              .LONG    0                                          ;
00000000  00154              .LONG    0                                          ;
00000000# 00158              .LONG    0[2]                                       ;
          00160 OBJECT_ESA:
                             .BLKB    255
          0025F              .BLKB    1
          00260 OBJECT_RSA:
                             .BLKB    255
          0035F              .BLKB    1
      02  00360 OBJECT_NAM:
                             .BYTE    2                                          ;
      60  00361              .BYTE    96                                         ;
      FF  00362              .BYTE    -1                                         ;
      00  00363              .BYTE    0                                          ;
00000000' 00364              .ADDRESS OBJECT_RSA                                 ;
      00  00368              .BYTE    0                                          ;
      00  00369              .BYTE    0                                          ;
      FF  0036A              .BYTE    -1                                         ;
      00  0036B              .BYTE    0                                          ;
00000000' 0036C              .ADDRESS OBJECT_ESA                                 ;
00000000' 00370              .ADDRESS OBJECT_RELATED_NAM                         ;
    0000# 00374              .WORD    0[8]                                       ;
    0000# 00384              .WORD    0[3]                                       ;
    0000# 0038A              .WORD    0[3]                                       ;
00000000  00390              .LONG    0                                          ;
00000000  00394              .LONG    0                                          ;
      00  00398              .BYTE    0                                          ;
      00  00399              .BYTE    0                                          ;
      00  0039A              .BYTE    0                                          ;
      00  0039B              .BYTE    0                                          ;
      00  0039C              .BYTE    0                                          ;
      00  0039D              .BYTE    0                                          ;
      00# 0039E              .BYTE    0[2]                                       ;
00000000  003A0              .LONG    0                                          ;
00000000  003A4              .LONG    0                                          ;
00000000  003A8              .LONG    0                                          ;
00000000  003AC              .LONG    0                                          ;
00000000  003B0              .LONG    0                                          ;
00000000  003B4              .LONG    0                                          ;
00000000# 003B8              .LONG    0[2]                                       ;
```

```
             00FF  003C0 OBJECT_SPEC:
                                  .WORD     255
           00  00  003C2          .BYTE     0, 0
        00000000'  003C4          .ADDRESS  OBJECT_SPEC+8
                   003C8          .BLKB     255
                   004C7          .BLKB     1
               03  004C8 OBJECT_FAB:
                                  .BYTE     3
               50  004C9          .BYTE     80
             0000  004CA          .WORD     0
         21000040  004CC          .LONG     553648192
         00000000  004D0          .LONG     0
         00000000  004D4          .LONG     0
         00000000  004D8          .LONG     0
             0000  004DC          .WORD     0
               01  004DE          .BYTE     1
               00  004DF          .BYTE     0
         00000000  004E0          .LONG     0
               00  004E4          .BYTE     0
               00  004E5          .BYTE     0
               00  004E6          .BYTE     0
               02  004E7          .BYTE     2
         00000000  004E8          .LONG     0
         00000000  004EC          .LONG     0
        00000000'  004F0          .ADDRESS  OBJECT_NAM
        00000000'  004F4          .ADDRESS  OBJECT_SPEC+8
        00000000'  004F8          .ADDRESS  P.AAA
               FF  004FC          .BYTE     -1
               04  004FD          .BYTE     4
             0000  004FE          .WORD     0
         00000000  00500          .LONG     0
             0000  00504          .WORD     0
               00  00506          .BYTE     0
               00  00507          .BYTE     0
         00000000  00508          .LONG     0
         00000000  0050C          .LONG     0
             0000  00510          .WORD     0
               00  00512          .BYTE     0
               00  00513          .BYTE     0
         00000000  00514          .LONG     0
               01  00518 OBJECT_RAB:
                                  .BYTE     1
               44  00519          .BYTE     68
             0000  0051A          .WORD     0
         00000400  0051C          .LONG     1024
         00000000  00520          .LONG     0
         00000000  00524          .LONG     0
            0000#  00528          .WORD     0[3]
             0000  0052E          .WORD     0
         00000000  00530          .LONG     0
             0000  00534          .WORD     0
               00  00536          .BYTE     0
               00  00537          .BYTE     0
             0000  00538          .WORD     0
             0000  0053A          .WORD     0
         00000000  0053C          .LONG     0
         00000000  00540          .LONG     0
```

```
                                        00000000  00544            .LONG    0                                           ;
                                        00000000  00548            .LONG    0                                           ;
                                              00  0054C            .BYTE    0                                           ;
                                              00  0054D            .BYTE    0                                           ;
                                              00  0054E            .BYTE    0                                           ;
                                              00  0054F            .BYTE    0                                           ;
                                        00000000  00550            .LONG    0                                           ;
                                        00000000' 00554            .ADDRESS OBJECT_FAB                                  ;
                                        00000000  00558            .LONG    0                                           ;

                                                                   .EXTRN   CDU$COLLECT_TABLE_BLOCKS
                                                                   .EXTRN   CDU$LOOKUP_CHILD
                                                                   .EXTRN   CDU$REPORT_RMS_ERROR
                                                                   .EXTRN   CLI$GET_VALUE, LIB$FREE_VM
                                                                   .EXTRN   LIB$GET_VM, CDU$FACILITY_STRING
                                                                   .EXTRN   CDU$GL_ROOT_NODE
                                                                   .EXTRN   CDU$GL_TABLE

                                                                   .PSECT   $CODE$,NOWRT,2

                                            007C 00000             .ENTRY   CDU$PREPARE_OBJECT_FILE, Save R2,R3,R4,R5,- ; 0875
                                                                            R6
                          50        04   AC  D0 00002              MOVL     CLD_FAB, R0                                 ; 0879
                          56        28   A0  D0 00006              MOVL     40(R0), R6                                  ;
                          50        01   A6  9A 0000A              MOVZBL   1(R6), R0                                   ; 0887
        0000'  CF         66             50  28 0000E              MOVC3    R0, (R6), OBJECT_RELATED_NAM                ;
                          50        02   A6  9A 00014              MOVZBL   2(R6), R0                                   ; 0888
        0000'  CF   04    B6             50  28 00018              MOVC3    R0, @4(R6), OBJECT_RELATED_RSA              ;
                                             04 0001F              RET                                                  ; 0892

; Routine Size:  32 bytes,    Routine Base:  $CODE$ + 0000
```

```
1 !++
1 ! Description:  This routine is called after all the CLD files have been
1 !               compiled.  It is responsible for creating and writing the
1 !               object file containing all of the generated table blocks,
1 !               along with related descriptive information.
1 !
1 ! Parameters:   None.
1 !
1 ! Returns:      Nothing.
1 !
1 ! Notes:
1 !--
1
1 GLOBAL ROUTINE cdu$write_object_file    : novalue
2 = BEGIN
2
2 local
2         status: long,
2         final_area: pointer;
2
2
2 ! Begin by creating the object file.  Get any value specified on the /OBJECT
2 ! qualifier to use as the spec for the object file.
2
2 cli$get_value(dtext('OBJECT'),object_spec);
2
2 ! Create and connect to the object file.  Any errors are fatal.
2
2 status = $create(fab=object_fab);
2 if not .status then
2         cdu$report_rms_error(msg(cdu$_openout),object_fab);
2 status = $connect(rab=object_rab);
2 if not .status then
2         cdu$report_rms_error(msg(cdu$_openout),object_rab);
2
2 ! Write the header records.
2
2 write_header_records();
2
2 ! Write the global symbol definition record.
2
2 write_global_symbol_record();
2
2 ! Allocate a large area to contain the final CLI table.  Collect all of the
2 ! table blocks into that area.
2
2 status = lib$get_vm(cdu$gl_table[vec_l_table_size], final_area);
2 check(.status, .status);
2 cdu$collect_table_blocks(.final_area);
2
2 ! Write the PSECT definition record.
2
2 write_psect_record();
2
2 ! Write the table blocks themselves.
2
2 write_table_records();
```

```
;   217         0950  2
;   218         0951  2  ! Write the records needed to define and store user routine addresses.
;   219         0952  2
;   220         0953  2  write_user_routine_records();
;   221         0954  2
;   222         0955  2  ! Write the end-of-module record.
;   223         0956  2
;   224         0957  2  write_eom_record();
;   225         0958  2
;   226         0959  2  return;
;   227         0960  2
;   228         0961  1  END;


                                                              .PSECT  $PLIT$,NOWRT,NOEXE,2

           00  00  54  43  45  4A  42  4F  00004 P.AAC:     .ASCII  \OBJECT\<0><0>                                    ;
                                 010E0006  0000C P.AAB:     .LONG   17694726                                          ;
                                 00000000' 00010            .ADDRESS P.AAC                                            ;

                                                            .EXTRN  SYS$CREATE, SYS$CONNECT

                                                            .PSECT  $CODE$,NOWRT,2

                                     001C  00000            .ENTRY  CDU$WRITE_OBJECT_FILE, Save R2,R3,R4          ; 0906
               54 00000000G   00      9E  00002            MOVAB   CDU$REPORT_RMS_ERROR, R4                          ;
               53     0000'   CF      9E  00009            MOVAB   OBJECT_FAB, R3                                    ;
               5E             04      C2  0000E            SUBL2   #4, SP                                            ;
                      FEF8    C3      9F  00011            PUSHAB  OBJECT_SPEC                                       ; 0917
                      0000'   CF      9F  00015            PUSHAB  P.AAB                                             ;
         00000000G   00       02      FB  00019            CALLS   #2, CLI$GET_VALUE                                 ;
                              53      DD  00020            PUSHL   R3                                                ; 0921
         00000000G   00       01      FB  00022            CALLS   #1, SYS$CREATE                                    ;
                     52       50      D0  00029            MOVL    R0, STATUS                                        ;
                     08       52      E8  0002C            BLBS    STATUS, 1$                                        ; 0922
                              53      DD  0002F            PUSHL   R3                                                ; 0923
                 001110A4     8F      DD  00031            PUSHL   #1118372                                          ;
                     64       02      FB  00037            CALLS   #2, CDU$REPORT_RMS_ERROR                          ;
                         50   A3      9F  0003A 1$:        PUSHAB  OBJECT_RAB                                        ; 0924
         00000000G   00       01      FB  0003D            CALLS   #1, SYS$CONNECT                                   ;
                     52       50      D0  00044            MOVL    R0, STATUS                                        ;
                     0C       52      E8  00047            BLBS    STATUS, 2$                                        ; 0925
                         50   A3      9F  0004A            PUSHAB  OBJECT_RAB                                        ; 0926
                 001110A4     8F      DD  0004D            PUSHL   #1118372                                          ;
                     64       02      FB  00053            CALLS   #2, CDU$REPORT_RMS_ERROR                          ;
             0000V   CF       00      FB  00056 2$:        CALLS   #0, WRITE_HEADER_RECORDS                          ; 0930
             0000V   CF       00      FB  0005B            CALLS   #0, WRITE_GLOBAL_SYMBOL_RECORD                    ; 0934
                              5E      DD  00060            PUSHL   SP                                                ; 0939
      7E 00000000G   00       10      C1  00062            ADDL3   #16, CDU$GL_TABLE, -(SP)                          ;
         00000000G   00       02      FB  0006A            CALLS   #2, LIB$GET_VM                                    ;
                     52       50      D0  00071            MOVL    R0, STATUS                                        ;
                     09       52      E8  00074            BLBS    STATUS, 3$                                        ; 0940
                              52      DD  00077            PUSHL   STATUS                                            ;
         00000000G   00       01      FB  00079            CALLS   #1, LIB$SIGNAL                                    ;
                              6E      DD  00080 3$:        PUSHL   FINAL_AREA                                        ; 0941
         00000000G   00       01      FB  00082            CALLS   #1, CDU$COLLECT_TABLE_BLOCKS                      ;
```

```
                              0000V  CF          00  FB 00089       CALLS   #0, WRITE_PSECT_RECORD                   ; 0945
                              0000V  CF          00  FB 0008E       CALLS   #0, WRITE_TABLE_RECORDS                  ; 0949
                              0000V  CF          00  FB 00093       CALLS   #0, WRITE_USER_ROUTINE_RECORDS           ; 0953
                              0000V  CF          00  FB 00098       CALLS   #0, WRITE_EOM_RECORD                     ; 0957
                                                     04 0009D       RET                                              ; 0961

; Routine Size:  158 bytes,    Routine Base:  $CODE$ + 0020
```

```
1 !++
1 ! Description:  This routine is responsible for writing the header records
1 !               in the object file.  We write the mandatory module record,
1 !               along with a language name record.
1 !
1 ! Parameters:   None.
1 !
1 ! Returns:      Nothing.
1 !
1 ! Notes:
1 !--
1
1 ROUTINE write_header_records    : novalue
2 = BEGIN
2
2 local
2         status: long,
2         hdr: block[256,byte],
2         variable_ptr: pointer,
2         child: ref node,
2         work_dsc: descriptor;
2
2
2 ! Set up the fixed portion of a module header record.
2
2 hdr[obj$b_rectyp] = obj$c_hdr;
2 hdr[mhd$b_hdrtyp] = mhd$c_mhd;
2 hdr[mhd$b_strlvl] = obj$c_strlvl;
2 hdr[mhd$w_recsiz] = obj$c_maxrecsiz;
2
2 ! Now we want to include the module name.  If there is a MODULE statement
2 ! in the CLD, use it.  Otherwise use the name of the object file.  While
2 ! we're at it, set up a pointer to the next available byte in the header.
2
2 child = cdu$lookup_child(.cdu$gl_root_node,node_k_module);
3 if .child nega 0 then (
3         ch$move(1+.child[node_b_text_length],child[node_b_text_length], hdr[mhd$b_namlng]);
3         variable_ptr = hdr[mhd$t_name] + .child[node_b_text_length];
3 ) else (
3         hdr[mhd$b_namlng] = .object_nam[nam$b_name];
3         ch$move(.object_nam[nam$b_name],.object_nam[nam$l_name], hdr[mhd$t_name]);
3         variable_ptr = hdr[mhd$t_name] + .object_nam[nam$b_name];
2 );
2
2 ! Now we want to include the module ident string.  If there is an IDENT
2 ! statement, then use it.  Otherwise use a string of "0-0".
2
2 child = cdu$lookup_child(.cdu$gl_root_node,node_k_ident);
3 if .child nega 0 then (
3         ch$move(1+.child[node_b_text_length],child[node_b_text_length], .variable_ptr);
3         variable_ptr = .variable_ptr + 1+.child[node_b_text_length];
3 ) else (
3         ch$move(4,ctext('0-0'), .variable_ptr);
3         variable_ptr = .variable_ptr + 4;
2 );
2
2 ! Finally, we want to include the current date and time.
```

```
2
2 build_descriptor(work_dsc,17,.variable_ptr);
2 status = $asctim(timbuf=work_dsc);
2 check(.status, .status);
2 variable_ptr = .variable_ptr + 17;
2
2 ! Write the module header into the object file.  Any error is fatal.
2
2 object_rab[rab$l_rbf] = hdr;
2 object_rab[rab$w_rsz] = .variable_ptr - hdr;
2 status = $put(rab=object_rab);
2 if not .status then
2         cdu$report_rms_error(msg(cdu$_writeerr),object_rab);
2
2 ! Set up the fixed portion of a language name record.
2
2 hdr[obj$b_rectyp] = obj$c_hdr;
2 hdr[mhd$b_hdrtyp] = mhd$c_lnm;
2
2 ! Move in our language name.
2
2 ch$move(.cdu$facility_string[len],.cdu$facility_string[ptr], hdr + 2);
2
2 ! Write the language name record in the object file.
2
2 object_rab[rab$w_rsz] = 2 + .cdu$facility_string[len];
2 status = $put(rab=object_rab);
2 if not .status then
2         cdu$report_rms_error(msg(cdu$_writeerr),object_rab);
2
2 return;
2
1 END;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

            30  2D  30  03  00014 P.AAD:                .ASCII  <3>\0-0\                                       ;

                                                        .EXTRN  SYS$ASCTIM, SYS$PUT

                                                        .PSECT  $CODE$,NOWRT,2

                                  0FFC 00000 WRITE_HEADER_RECORDS:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11   ; 0974
          5B 00000000G   00   9E 00002                  MOVAB   SYS$PUT, R11                           ;
          5A 00000000G   00   9E 00009                  MOVAB   CDU$LOOKUP_CHILD, R10                  ;
          59      0000'  CF   9E 00010                  MOVAB   OBJECT_RAB, R9                         ;
          5E      FEF8   CE   9E 00015                  MOVAB   -264(SP), SP                           ;
                    08   AE   B4 0001A                  CLRW    HDR                                    ; 0987
                    0A   AE   94 0001D                  CLRB    HDR+2                                  ; 0989
    0B    AE      0800   8F   B0 00020                  MOVW    #2048, HDR+3                           ; 0990
                         03   DD 00026                  PUSHL   #3                                     ; 0996
             00000000G   00   DD 00028                  PUSHL   CDU$GL_ROOT_NODE                       ;
          6A             02   FB 0002E                  CALLS   #2, CDU$LOOKUP_CHILD                   ;
          57             50   D0 00031                  MOVL    R0, CHILD                              ;
```

```
                              19  13 00034          BEQL    1$                                  ; 0997
                 50        10 A7  9A 00036          MOVZBL  16(CHILD), R0                       ; 0998
                              50  D6 0003A          INCL    R0                                  ;
0D  AE        10 A7           50  28 0003C          MOVC3   R0, 16(CHILD), HDR+5                ;
                 50        0E AE  9E 00042          MOVAB   HDR+6, R0                           ; 0999
                 58        10 A7  9A 00046          MOVZBL  16(CHILD), VARIABLE_PTR             ;
                 58           50  C0 0004A          ADDL2   R0, VARIABLE_PTR                    ;
                              15  11 0004D          BRB     2$                                  ; 0997
                 56      FE83 C9  9A 0004F 1$:      MOVZBL  OBJECT_NAM+59, R6                   ; 1001
              0D AE           56  90 00054          MOVB    R6, HDR+5                           ;
0E  AE      FE94 D9           56  28 00058          MOVC3   R6, @OBJECT_NAM+76, HDR+6           ; 1002
                 58     0E AE 46  9E 0005F          MOVAB   HDR+6[R6], VARIABLE_PTR             ; 1003
                              02  DD 00064 2$:      PUSHL   #2                                  ; 1009
                    00000000G 00  DD 00066          PUSHL   CDU$GL_ROOT_NODE                    ;
                 6A           02  FB 0006C          CALLS   #2, CDU$LOOKUP_CHILD                ;
                 57           50  D0 0006F          MOVL    R0, CHILD                           ;
                              14  13 00072          BEQL    3$                                  ; 1010
                 56        10 A7  9A 00074          MOVZBL  16(CHILD), R6                       ; 1011
                 50        01 A6  9E 00078          MOVAB   1(R6), R0                           ;
    68        10 A7           50  28 0007C          MOVC3   R0, 16(CHILD), (VARIABLE_PTR)       ;
                 58     01 A6 48  9E 00081          MOVAB   1(R6)[VARIABLE_PTR], VARIABLE_PTR   ; 1012
                              05  11 00086          BRB     4$                                  ; 1010
                 8B     0000' CF  D0 00088 3$:      MOVL    P.AAD, (VARIABLE_PTR)+              ; 1014
                 6E           11  D0 0008D 4$:      MOVL    #17, WORK_DSC                       ; 1020
              04 AE           58  D0 00090          MOVL    VARIABLE_PTR, WORK_DSC+4            ;
                              7E  7C 00094          CLRQ    -(SP)                               ; 1021
                           08 AE  9F 00096          PUSHAB  WORK_DSC                            ;
                              7E  D4 00099          CLRL    -(SP)                               ;
       00000000G 00           04  FB 0009B          CALLS   #4, SYS$ASCTIM                      ;
                 57           50  D0 000A2          MOVL    R0, STATUS                          ;
                 09           57  E8 000A5          BLBS    STATUS, 5$                          ; 1022
                              57  DD 000A8          PUSHL   STATUS                              ;
       00000000G 00           01  FB 000AA          CALLS   #1, LIB$SIGNAL                      ;
                 58           11  C0 000B1 5$:      ADDL2   #17, VARIABLE_PTR                   ; 1023
              28 A9        08 AE  9E 000B4          MOVAB   HDR, OBJECT_RAB+40                  ; 1027
                 50        08 AE  9E 000B9          MOVAB   HDR, R0                             ; 1028
22  A9           58           50  A3 000BD          SUBW3   R0, VARIABLE_PTR, OBJECT_RAB+34     ;
                              59  DD 000C2          PUSHL   R9                                  ; 1029
                 6B           01  FB 000C4          CALLS   #1, SYS$PUT                         ;
                 57           50  D0 000C7          MOVL    R0, STATUS                          ;
                 0F           57  E8 000CA          BLBS    STATUS, 6$                          ; 1030
                              59  DD 000CD          PUSHL   R9                                  ; 1031
                    001110D4  8F  DD 000CF          PUSHL   #1118420                            ;
       00000000G 00           02  FB 000D5          CALLS   #2, CDU$REPORT_RMS_ERROR            ;
              08 AE      0100 8F  B0 000DC 6$:      MOVW    #256, HDR                           ; 1035
                 56 00000000G 00  3C 000E2          MOVZWL  CDU$FACILITY_STRING, R6             ; 1040
                 50 00000000G 00  D0 000E9          MOVL    CDU$FACILITY_STRING+4, R0           ;
0A  AE           60           56  28 000F0          MOVC3   R6, (R0), HDR+2                     ;
22  A9           56           02  A1 000F5          ADDW3   #2, R6, OBJECT_RAB+34               ; 1044
                              59  DD 000FA          PUSHL   R9                                  ; 1045
                 6B           01  FB 000FC          CALLS   #1, SYS$PUT                         ;
                 57           50  D0 000FF          MOVL    R0, STATUS                          ;
                 0F           57  E8 00102          BLBS    STATUS, 7$                          ; 1046
                              59  DD 00105          PUSHL   R9                                  ; 1047
                    001110D4  8F  DD 00107          PUSHL   #1118420                            ;
       00000000G 00           02  FB 0010D          CALLS   #2, CDU$REPORT_RMS_ERROR            ;
                                  04 00114 7$:      RET                                         ; 1051
```

```
; Routine Size:  277 bytes,    Routine Base:  $CODE$ + 00BE
```

```
1 !++
1 ! Description:  This routine is responsible for writing a global symbol
1 !               directory record to define the global symbol naming the
1 !               table.  This name is used in CLI calls to reference
1 !               this table after it is linked with an image.
1 !
1 ! Parameters:   None.
1 !
1 ! Returns:      Nothing.
1 !
1 ! Notes:
1 !--
1
1 ROUTINE write_global_symbol_record                : novalue
2 = BEGIN
2
2 local
2         status: long,
2         gsd: block[256,byte],
2         child: ref node;
2
2 bind
2         gsd_sym = gsd + 1: block[,byte];
2
2
2 ! Set up the fixed portion of the record.
2
2 gsd[obj$b_rectyp] = obj$c_gsd;
2 gsd_sym[sdf$b_gsdtyp] = gsd$c_sym;
2 gsd_sym[sdf$b_datyp] = 0;
2 gsd_sym[sdf$w_flags] = gsy$m_def + gsy$m_rel;
2 gsd_sym[sdf$b_psindx] = 0;
2 gsd_sym[sdf$l_value] = 0;
2
2 ! Now we want the module name as the symbol.  If there is a MODULE statement
2 ! in the CLD, use it.  Otherwise use the name of the object file.
2
2 child = cdu$lookup_child(.cdu$gl_root_node,node_k_module);
2 if .child neqa 0 then
2         ch$move(1+.child[node_b_text_length],child[node_b_text_length],
2                 gsd_sym[sdf$b_namlng])
3 else (
3         gsd_sym[sdf$b_namlng] = .object_nam[nam$b_name];
3         ch$move(.object_nam[nam$b_name],.object_nam[nam$l_name],
3                 gsd_sym[sdf$t_name]);
2 );
2
2 ! Write the record into the object file.  Any error is fatal.
2
2 object_rab[rab$l_rbf] = gsd;
2 object_rab[rab$w_rsz] = 1 + 9 + 1+.gsd_sym[sdf$b_namlng];
2 status = $put(rab=object_rab);
2 if not .status then
2         cdu$report_rms_error(msg(cdu$_writeerr),object_rab);
2
2 return;
2
```

```
;  378          1109  1 END;



                                                                  003C 00000 WRITE_GLOBAL_SYMBOL_RECORD:
                                                                                         .WORD     Save R2,R3,R4,R5                                ; 1065
                                                5E       FF00   CE   9E 00002            MOVAB     -256(SP), SP                                    ;
                                                6E       0101   8F   B0 00007            MOVW      #257, GSD                                       ; 1079
                                                           02   AE   94 0000C            CLRB      GSD_SYM+1                                       ; 1081
                                           03   AE              0A   B0 0000F            MOVW      #10, GSD_SYM+2                                  ; 1082
                                                           05   AE   94 00013            CLRB      GSD_SYM+4                                       ; 1083
                                                           06   AE   D4 00016            CLRL      GSD_SYM+5                                       ; 1084
                                                                03   DD 00019            PUSHL     #3                                              ; 1089
                                                    00000000G   00   DD 0001B            PUSHL     CDU$GL_ROOT_NODE                                ;
                                    00000000G   00              02   FB 00021            CALLS     #2, CDU$LOOKUP_CHILD                            ;
                                                                50   D5 00028            TSTL      CHILD                                           ; 1090
                                                                0E   13 0002A            BEQL      1$                                              ;
                                                51         10   A0   9A 0002C            MOVZBL    16(CHILD), R1                                   ; 1091
                                                                51   D6 00030            INCL      R1                                              ;
                        0A    AE           10   A0              51   28 00032            MOVC3     R1, 16(CHILD), GSD_SYM+9                        ; 1092
                                                                12   11 00038            BRB       2$                                              ;
                                           0A   AE      0000'   CF   90 0003A  1$:       MOVB      OBJECT_NAM+59, GSD_SYM+9                        ; 1094
                                                50      0000'   CF   9A 00040            MOVZBL    OBJECT_NAM+59, R0                               ; 1095
                        0B    AE        0000'   DF              50   28 00045            MOVC3     R0, @OBJECT_NAM+76, GSD_SYM+10                  ; 1096
                                        0000'   CF              6E   9E 0004C  2$:       MOVAB     GSD, OBJECT_RAB+40                              ; 1101
                                        0000'   CF         0A   AE   9B 00051            MOVZBW    GSD_SYM+9, OBJECT_RAB+34                        ; 1102
                                        0000'   CF              0B   A0 00057            ADDW2     #11, OBJECT_RAB+34                              ;
                                                        0000'   CF   9F 0005C            PUSHAB    OBJECT_RAB                                      ; 1103
                                    00000000G   00              01   FB 00060            CALLS     #1, SYS$PUT                                     ;
                                                11              50   E8 00067            BLBS      STATUS, 3$                                      ; 1104
                                                        0000'   CF   9F 0006A            PUSHAB    OBJECT_RAB                                      ; 1105
                                                     001110D4   8F   DD 0006E            PUSHL     #1118420                                        ;
                                    00000000G   00              02   FB 00074            CALLS     #2, CDU$REPORT_RMS_ERROR                        ;
                                                                     04 0007B  3$:       RET                                                       ; 1109

; Routine Size:  124 bytes,    Routine Base:  $CODE$ + 01D3
```

```
1 !++
1 ! Description:  This routine is responsible for writing the psect definition
1 !               record, which defines the psect in which all the blocks reside.
1 !
1 ! Parameters:   None.
1 !
1 ! Returns:      Nothing.
1 !
1 ! Notes:
1 !--
1
1 ROUTINE write_psect_record             : novalue
2 = BEGIN
2
2 local
2         status: long,
2         gsd: block[256,byte];
2
2 bind
2         gsd_psc = gsd + 1: block[,byte];
2
2
2 ! Set up the fixed portion of the psect record.  We get the psect size out
2 ! of the primary vector block.
2
2 gsd[obj$b_rectyp] = obj$c_gsd;
2 gsd_psc[gps$b_gsdtyp] = gsd$c_psc;
2 gsd_psc[gps$b_align] = 2;
2 gsd_psc[gps$w_flags] = gps$m_pic + gps$m_rel + gps$m_rd;
2 gsd_psc[gps$l_alloc] = .cdu$gl_table[vec_l_table_size];
2
2 ! Now we want the psect name.
2
3 begin
3 bind
3         name = ctext('CLI$TABLES'): vector[,byte];
3
3 ch$move(1+.name[0],name[0], gsd_psc[gps$b_namlng]);
2 end;
2
2 ! Write the psect definition record into the object file.  Errors are fatal.
2
2 object_rab[rab$l_rbf] = gsd;
2 object_rab[rab$w_rsz] = 1 + 8 + 1+.gsd_psc[gps$b_namlng];
2 status = $put(rab=object_rab);
2 if not .status then
2         cdu$report_rms_error(msg(cdu$_writeerr),object_rab);
2
2 return;
2
1 END;


                                                              .PSECT  $PLIT$,NOWRT,NOEXE,2

53  45  4C  42  41  54  24  49  4C  43  0A  00018 P.AAE:  .ASCII  <10>\CLI$TABLES\                    ;
```

```
                                                        NAME=                P.AAE

                                                                .PSECT  $CODE$,NOWRT,2

                                           007C 00000 WRITE_PSECT_RECORD:
                                                                .WORD   Save R2,R3,R4,R5,R6                              ; 1121
                      56       0000'   CF   9E 00002            MOVAB   OBJECT_RAB+34, R6                                ;
                      5E       FF00    CE   9E 00007            MOVAB   -256(SP), SP                                     ;
                      6E               01   B0 0000C            MOVW    #1, GSD                                          ; 1135
                02    AE               02   90 0000F            MOVB    #2, GSD_PSC+1                                    ; 1137
                03    AE         89    8F   9B 00013            MOVZBW  #137, GSD_PSC+2                                  ; 1138
                      50  00000000G    00   D0 00018            MOVL    CDU$GL_TABLE, R0                                 ; 1139
                05    AE         10    A0   D0 0001F            MOVL    16(R0), GSD_PSC+4                                ;
                      50       0000'   CF   9A 00024            MOVZBL  NAME, R0                                         ; 1147
                                       50   D6 00029            INCL    R0                                               ;
        09    AE      0000'   CF       50   28 0002B            MOVC3   R0, NAME, GSD_PSC+8                              ;
                06    A6               6E   9E 00032            MOVAB   GSD, OBJECT_RAB+40                               ; 1152
                      66         09    AE   9B 00036            MOVZBW  GSD_PSC+8, OBJECT_RAB+34                         ; 1153
                      66               0A   A0 0003A            ADDW2   #10, OBJECT_RAB+34                               ;
                                 DE    A6   9F 0003D            PUSHAB  OBJECT_RAB                                       ; 1154
           00000000G  00               01   FB 00040            CALLS   #1, SYS$PUT                                      ;
                      10               50   E8 00047            BLBS    STATUS, 1$                                       ; 1155
                                 DE    A6   9F 0004A            PUSHAB  OBJECT_RAB                                       ; 1156
                           001110D4    8F   DD 0004D            PUSHL   #1118420                                         ;
           00000000G  00               02   FB 00053            CALLS   #2, CDU$REPORT_RMS_ERROR                         ;
                                            04 0005A 1$:        RET                                                      ; 1160

; Routine Size:  91 bytes,    Routine Base:  $CODE$ + 024F
```

```
1 !++
1 ! Description:  This routine is called to write a sequence of TIR records
1 !               containing the table blocks.  The blocks are packed
1 !               together, resulting in a minimum number of records.
1 !
1 ! Parameters:   None.
1 !
1 ! Returns:      Nothing.
1 !
1 ! Notes:        We assume the table blocks have been collected into a final,
1 !               contiguous area.
1 !--
1
1 ROUTINE write_table_records     : novalue
2 = BEGIN
2
2 local
2         status: long,
2         tir: block[obj$c_maxrecsiz,byte],
2         table_offset: long,
2         command: pointer,
2         command_length: long;
2
2
2 ! Initialize the type byte of the TIR record.
2
2 tir[obj$b_rectyp] = obj$c_tir;
2
2 ! Write out the following sequence of TIR commands, which will set the
2 ! location counter to the beginning of the psect.
2 !
2 !       stack address of beginning of psect
2 !       set location counter
2 !
2 ! Any error is fatal.
2
2 tir[1,0,8,0] = tir$c_sta_pb;
2 tir[2,0,8,0] = 0;
2 tir[3,0,8,0] = 0;
2 tir[4,0,8,0] = tir$c_ctl_setrb;
2 object_rab[rab$l_rbf] = tir;
2 object_rab[rab$w_rsz] = 1 + 3 + 1;
2 status = $put(rab=object_rab);
2 if not .status then
2         cdu$report_rms_error(msg(cdu$_writeerr),object_rab);
2
2 ! Sit in a loop, going through once for each TIR record.  The table offset
2 ! pointer will advance along the CLI table as we write it out.
2
2 table_offset = 0;
3 do (
3
3         ! Initialize the command pointer, which will advance along the TIR
3         ! record, to point past the type byte.
3
3         command = tir + 1;
3
;  489        1218  3          ! Each TIR record contains a sequence of Store Immediate commands.
;  490        1219  3          ! Loop once for each command.
;  491        1220  3
;  492        1221  4          incru i from 1 to obj$c_maxrecsiz / 129 do (
;  493        1222  4
;  494        1223  4                  ! The Store Immediate command is the negative of the length
;  495        1224  4                  ! of the bytes being stored.  That's 128 bytes unless we are
;  496        1225  4                  ! at the end of the table.
;  497        1226  4
;  498        1227  4                  command_length = minu(128, .cdu$gl_table[vec_l_table_size]-.table_offset);
;  499        1228  4                  command[0,0,8,1] = -.command_length;
;  500        1229  4
;  501        1230  4                  ! Copy the table bytes following the Store Immediate
;  502        1231  4                  ! command.
;  503        1232  4
;  504        1233  4                  ch$move(.command_length,.cdu$gl_table+.table_offset, command[1,0,0,0]);
;  505        1234  4
;  506        1235  4                  ! Advance the table offset and the command pointer.
;  507        1236  4
;  508        1237  4                  table_offset = .table_offset + .command_length;
;  509        1238  4                  command = .command + 1+.command_length;
;  510        1239  4
;  511        1240  4                  ! If we've finished copying the table, then get out of this
;  512        1241  4                  ! loop.
;  513        1242  4
;  514        1243  4                  if .table_offset eqlu .cdu$gl_table[vec_l_table_size] then exitloop;
;  515        1244  3          );
;  516        1245  3
;  517        1246  3          ! Write the TIR record.  Any error is fatal.
;  518        1247  3
;  519        1248  3          object_rab[rab$w_rsz] = .command - tir;
;  520        1249  3          status = $put(rab=object_rab);
;  521        1250  3          if not .status then
;  522        1251  3                  cdu$report_rms_error(msg(cdu$_writeerr),object_rab);
;  523        1252  3
;  524        1253  3          ! Loop until we have written the entire table.
;  525        1254  3
;  526        1255  2  ) until .table_offset eqlu .cdu$gl_table[vec_l_table_size];
;  527        1256  2
;  528        1257  2  return;
;  529        1258  2
;  530        1259  1  END;



                                   0FFC 00000 WRITE_TABLE_RECORDS:
                                                      .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 1174
           5E     F7FC    CE  9E 00002               MOVAB   -2052(SP), SP                                 ;
     04    AE     0402    8F  3C 00007               MOVZWL  #1026, TIR                                    ; 1187
     08    AE       50    8F  90 0000D               MOVB    #80, TIR+4                                    ; 1200
  0000'    CF       04    AE  9E 00012               MOVAB   TIR, OBJECT_RAB+40                            ; 1201
  0000'    CF             05  B0 00018               MOVW    #5, OBJECT_RAB+34                             ; 1202
                  0000'   CF  9F 0001D               PUSHAB  OBJECT_RAB                                    ; 1203
000000000G 00             01  FB 00021               CALLS   #1, SYS$PUT                                   ;
           6E             50  D0 00028               MOVL    R0, STATUS                                    ;
```

```
                              11          6E  E8 0002B         BLBS    STATUS, 1$                                         ; 1204
                                    0000'  CF  9F 0002E         PUSHAB  OBJECT_RAB                                         ; 1205
                                 001110D4  8F  DD 00032         PUSHL   #1118420                                           ;
                   00000000G  00           02  FB 00038         CALLS   #2, CDU$REPORT_RMS_ERROR                           ;
                                           59  D4 0003F  1$:    CLRL    TABLE_OFFSET                                       ; 1210
                              57 00000000G 00  D0 00041         MOVL    CDU$GL_TABLE, R7                                   ; 1227
                              5A        10 A7  9E 00048         MOVAB   16(R7), R10                                        ;
                              56        05 AE  9E 0004C  2$:    MOVAB   TIR+1, COMMAND                                     ; 1216
                              5B           01  D0 00050         MOVL    #1, I                                              ; 1221
                 50           6A           59  C3 00053  3$:    SUBL3   TABLE_OFFSET, (R10), R0                            ; 1227
                   00000080   8F           50  D1 00057         CMPL    R0, #128                                           ;
                                           04  1B 0005E         BLEQU   4$                                                 ;
                              50        80 8F  9A 00060         MOVZBL  #128, R0                                           ;
                              58           50  D0 00064  4$:    MOVL    R0, COMMAND_LENGTH                                 ;
                              66           58  8E 00067         MNEGB   COMMAND_LENGTH, (COMMAND)                          ; 1228
          01     A6         6947           58  28 0006A         MOVC3   COMMAND_LENGTH, (TABLE_OFFSET)[R7], -              ; 1233
                                                                        1(COMMAND)                                         ;
                              59           58  C0 00070         ADDL2   COMMAND_LENGTH, TABLE_OFFSET                       ; 1237
                              56     01 AB46   9E 00073         MOVAB   1(COMMAND_LENGTH)[COMMAND], COMMAND                ; 1238
                              6A           59  D1 00078         CMPL    TABLE_OFFSET, (R10)                                ; 1243
                                           07  13 0007B         BEQL    5$                                                 ;
                                           5B  D6 0007D         INCL    I                                                  ; 1221
                              0F           5B  D1 0007F         CMPL    I, #15                                             ;
                                           CF  1B 00082         BLEQU   3$                                                 ;
                              50        04 AE  9E 00084  5$:    MOVAB   TIR, R0                                            ; 1248
        0000'    CF           56           50  A3 00088         SUBW3   R0, COMMAND, OBJECT_RAB+34                         ;
                                    0000'  CF  9F 0008E         PUSHAB  OBJECT_RAB                                         ; 1249
                   00000000G  00           01  FB 00092         CALLS   #1, SYS$PUT                                        ;
                              6E           50  D0 00099         MOVL    R0, STATUS                                         ;
                              11           6E  E8 0009C         BLBS    STATUS, 6$                                         ; 1250
                                    0000'  CF  9F 0009F         PUSHAB  OBJECT_RAB                                         ; 1251
                                 001110D4  8F  DD 000A3         PUSHL   #1118420                                           ;
                   00000000G  00           02  FB 000A9         CALLS   #2, CDU$REPORT_RMS_ERROR                           ;
                              57 00000000G 00  D0 000B0  6$:    MOVL    CDU$GL_TABLE, R7                                   ; 1255
                              5A        10 A7  9E 000B7         MOVAB   16(R7), R10                                        ;
                              6A           59  D1 000BB         CMPL    TABLE_OFFSET, (R10)                                ;
                                           8C  12 000BE         BNEQ    2$                                                 ;
                                               04 000C0         RET                                                        ; 1259

; Routine Size:  193 bytes,    Routine Base:  $CODE$ + 02AA
```

```
1 !++
1 ! Description:  This routine is called to write out the records needed to
1 !               declare and store the references to user routines which
1 !               handle verbs.  These routines are specified by ROUTINE
1 !               clauses in the CLD and must be resolved by the Linker.
1 !
1 !               The task is accomplished by traversing all of the table
1 !               blocks looking for command blocks which specify user
1 !               routines.
1 !
1 ! Parameters:   None.
1 !
1 ! Returns:      Nothing.
1 !
1 ! Notes:
1 !--
1
1 ROUTINE write_user_routine_records              : novalue
2 = BEGIN
2
2 local
2         status: long,
2         a_block: pointer,
2         obj: block[256,byte];
2 bind
2         gsd_sym = obj + 1: block[,byte];
2
2
2 ! Loop through each of the table blocks, one at at a time.  When a command
2 ! block with a user routine handler is encoutered, then we have to do some
2 ! work.
2
2 a_block = .cdu$gl_table;
3 while .a_block lssa .cdu$gl_table + .cdu$gl_table[vec_l_table_size] do (
3
3         if .a_block[vec_b_type] eqlu block_k_command then if
4            .a_block[cmd_b_handler] eqlu cmd_k_user       then (
4
4                 bind
4                         symbol = .a_block + .a_block[cmd_w_image]+4: vector[,byte];
4
4                 ! First we must generate a GSD record to declare the user
4                 ! routine address.  The symbol for this address is stored in
4                 ! the command block at the offset specified by the image BRO
4                 ! (plus four for the reference longword).
4
4                 ! Set up the fixed portion of the record.
4
4                 obj[obj$b_rectyp] = obj$c_gsd;
4                 gsd_sym[srf$b_gsdtyp] = gsd$c_sym;
4                 gsd_sym[srf$b_datyp] = 0;
4                 gsd_sym[srf$w_flags] = 0;
4
4                 ! Move the symbol into the record.
4
4                 ch$move(1+.symbol[0],symbol[0], gsd_sym[srf$b_namlng]);
4
4                       ! Write the record into the object file.  Any error is fatal.
4
4                       object_rab[rab$l_rbf] = obj;
4                       object_rab[rab$w_rsz] = 1 + 4 + 1+.symbol[0];
4                       status = $put(rab=object_rab);
4                       if not .status then
4                               cdu$report_rms_error(msg(cdu$_writeerr),object_rab);
4
4                       ! Now we have to write a TIR record with the following sequence
4                       ! of commands to store the user routine address in the command
4                       ! block.
4                       !
4                       !       stack address of user routine reference longword
4                       !       set location counter
4                       !       stack address of user routine
4                       !       store PIC data reference
4
4                       ! Build the fixed portion of the commands.
4
4                       obj[obj$b_rectyp] = obj$c_tir;
4                       obj[1,0,8,0] = tir$c_sta_pl;
4                       obj[2,0,8,0] = 0;
4                       obj[3,0,32,0] = .a_block - .cdu$gl_table + .a_block[cmd_w_image];
4                       obj[7,0,8,0] = tir$c_ctl_setrb;
4                       obj[8,0,8,0] = tir$c_sta_gbl;
4
4                       ! Move the symbol in as the operand of the stack global.
4
4                       ch$move(1+.symbol[0],symbol[0], obj[9,0,0,0]);
4
4                       ! Finish the command sequence.
4
4                       obj[9 + 1+.symbol[0],0,8,0] = tir$c_sto_pidr;
4
4                       ! Write the record into the object file.  Any error is fatal.
4
4                       object_rab[rab$w_rsz] = 1 + 6 + 1 + 1+1+.symbol[0] + 1;
4                       status = $put(rab=object_rab);
4                       if not .status then
4                               cdu$report_rms_error(msg(cdu$_writeerr),object_rab);
3               );
3
3               ! Move on to the next table block.
3
3               a_block = .a_block + .a_block[vec_w_size];
2       );
2
2       return;
2
1       END;



                                    OFFC 00000 WRITE_USER_ROUTINE_RECORDS:
                                                    .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 1277
```

```
                        5B      0000'   CF  9E 00002        MOVAB   OBJECT_RAB+34, R11                          ;
                        5E       FF00   CE  9E 00007        MOVAB   -256(SP), SP                                ;
                        56  00000000G   00  D0 0000C        MOVL    CDU$GL_TABLE, A_BLOCK                       ; 1292
                        50  00000000G   00  D0 00013  1$:   MOVL    CDU$GL_TABLE, R0                            ; 1293
                        50         10   A0  C0 0001A        ADDL2   16(R0), R0                                  ;
                        50              56  D1 0001E        CMPL    A_BLOCK, R0                                 ;
                                        01  1F 00021        BLSSU   2$                                          ;
                                            04 00023        RET                                                 ;
                        02         02   A6  91 00024  2$:   CMPB    2(A_BLOCK), #2                              ; 1295
                                        04  12 00028        BNEQ    3$                                          ;
                        02         14   A6  91 0002A        CMPB    20(A_BLOCK), #2                             ; 1296
                                        03  13 0002E  3$:   BEQL    4$                                          ;
                                      0095  31 00030        BRW     6$                                          ;
                        57         1A   A6  3C 00033  4$:   MOVZWL  26(A_BLOCK), R7                             ; 1299
                        58         04 A746  9E 00037        MOVAB   4(R7)[A_BLOCK], R8                          ;
                        6E       0101   8F  B0 0003C        MOVW    #257, OBJ                                   ; 1308
                                   02   AE  94 00041        CLRB    GSD_SYM+1                                   ; 1310
                                   03   AE  B4 00044        CLRW    GSD_SYM+2                                   ; 1311
                        59              68  9A 00047        MOVZBL  (R8), R9                                    ; 1315
                                        59  D6 0004A        INCL    R9                                          ;
    05    AE            68              59  28 0004C        MOVC3   R9, (R8), GSD_SYM+4                         ;
                  06    AB              6E  9E 00051        MOVAB   OBJ, OBJECT_RAB+40                          ; 1319
                        6B              68  9B 00055        MOVZBW  (R8), OBJECT_RAB+34                         ; 1320
                        6B              06  A0 00058        ADDW2   #6, OBJECT_RAB+34                           ;
                                   DE   AB  9F 0005B        PUSHAB  OBJECT_RAB                                  ; 1321
             00000000G  00              01  FB 0005E        CALLS   #1, SYS$PUT                                 ;
                        5A              50  D0 00065        MOVL    R0, STATUS                                  ;
                        10              5A  E8 00068        BLBS    STATUS, 5$                                  ; 1322
                                   DE   AB  9F 0006B        PUSHAB  OBJECT_RAB                                  ; 1323
                             001110D4   8F  DD 0006E        PUSHL   #1118420                                    ;
             00000000G  00              02  FB 00074        CALLS   #2, CDU$REPORT_RMS_ERROR                    ;
                        6E       0602   8F  B0 0007B  5$:   MOVW    #1538, OBJ                                  ; 1336
                                   02   AE  94 00080        CLRB    OBJ+2                                       ; 1338
          50            56  00000000G   00  C3 00083        SUBL3   CDU$GL_TABLE, A_BLOCK, R0                   ; 1339
    03    AE            50              57  C1 0008B        ADDL3   R7, R0, OBJ+3                               ;
                  07    AE         50   8F  9B 00090        MOVZBW  #80, OBJ+7                                  ; 1340
    09    AE            68              59  28 00095        MOVC3   R9, (R8), OBJ+9                             ; 1345
                        50              68  9A 0009A        MOVZBL  (R8), R0                                    ; 1349
                   0A AE40              1B  90 0009D        MOVB    #27, OBJ+10[R0]                             ;
                        6B              68  9B 000A2        MOVZBW  (R8), OBJECT_RAB+34                         ; 1353
                        6B              0B  A0 000A5        ADDW2   #11, OBJECT_RAB+34                          ;
                                   DE   AB  9F 000A8        PUSHAB  OBJECT_RAB                                  ; 1354
             00000000G  00              01  FB 000AB        CALLS   #1, SYS$PUT                                 ;
                        5A              50  D0 000B2        MOVL    R0, STATUS                                  ;
                        10              5A  E8 000B5        BLBS    STATUS, 6$                                  ; 1355
                                   DE   AB  9F 000B8        PUSHAB  OBJECT_RAB                                  ; 1356
                             001110D4   8F  DD 000BB        PUSHL   #1118420                                    ;
             00000000G  00              02  FB 000C1        CALLS   #2, CDU$REPORT_RMS_ERROR                    ;
                        50              66  3C 000C8  6$:   MOVZWL  (A_BLOCK), R0                               ; 1361
                        56              50  C0 000CB        ADDL2   R0, A_BLOCK                                 ;
                                      FF42  31 000CE        BRW     1$                                          ; 1293
                                            04 000D1        RET                                                 ; 1366

; Routine Size:  210 bytes,    Routine Base:  $CODE$ + 036B
;  640          1367  1 !++
;  641          1368  1 ! Description:  This routine is responsible for writing the end-of-module
;  642          1369  1 !               record at the end of the object file.
;  643          1370  1 !
;  644          1371  1 ! Parameters:   None.
;  645          1372  1 !
;  646          1373  1 ! Returns:      Nothing.
;  647          1374  1 !
;  648          1375  1 ! Notes:
;  649          1376  1 !--
;  650          1377  1
;  651          1378  1 ROUTINE write_eom_record        : novalue
;  652          1379  2 = BEGIN
;  653          1380  2
;  654          1381  2 local
;  655          1382  2         status: long,
;  656          1383  2         eom: block[256,byte];
;  657          1384  2
;  658          1385  2 ! Format the end-of-module record.
;  659          1386  2
;  660          1387  2 eom[obj$b_rectyp] = obj$c_eom;
;  661          1388  2 eom[eom$b_comcod] = 0;
;  662          1389  2
;  663          1390  2 ! Write the record.  All errors are fatal.
;  664          1391  2
;  665          1392  2 object_rab[rab$l_rbf] = eom;
;  666          1393  2 object_rab[rab$w_rsz] = 2;
;  667          1394  2 status = $put(rab=object_rab);
;  668          1395  2 if not .status then
;  669          1396  2         cdu$report_rms_error(msg(cdu$_writeerr),object_rab);
;  670          1397  2
;  671          1398  2 return;
;  672          1399  2
;  673          1400  1 END;



                                       0004 00000 WRITE_EOM_RECORD:
                                                         .WORD   Save R2                                    ; 1378
                  52       0000'   CF   9E 00002         MOVAB   OBJECT_RAB, R2                             ;
                  5E       FF00    CE   9E 00007         MOVAB   -256(SP), SP                               ;
                  6E               03   B0 0000C         MOVW    #3, EOM                                    ; 1387
            28    A2               6E   9E 0000F         MOVAB   EOM, OBJECT_RAB+40                         ; 1392
            22    A2               02   B0 00013         MOVW    #2, OBJECT_RAB+34                          ; 1393
                                   52   DD 00017         PUSHL   R2                                         ; 1394
     00000000G    00               01   FB 00019         CALLS   #1, SYS$PUT                                ;
                  0F               50   E8 00020         BLBS    STATUS, 1$                                 ; 1395
                                   52   DD 00023         PUSHL   R2                                         ; 1396
                        001110D4   8F   DD 00025         PUSHL   #1118420                                   ;
     00000000G    00               02   FB 0002B         CALLS   #2, CDU$REPORT_RMS_ERROR                   ;
                                        04 00032 1$:     RET                                                ; 1400

; Routine Size:  51 bytes,    Routine Base:  $CODE$ + 043D
```

```
                                                                E  7
OBJECT                                                         15-Sep-1984 23:45:30     VAX-11 Bliss-32 V4.0-742                Page  26
V04-000                                                        14-Sep-1984 11:58:25     DISK$VMSMASTER:[CDU.SRC]OBJECT.B32;1      (11)

;  674           1401  1 END
;  675           1402  0 ELUDOM



                                                                            .EXTRN  LIB$SIGNAL

;                                          PSECT SUMMARY
;
;       Name                           Bytes                             Attributes
;
;   $OWN$                                  1372  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   $PLIT$                                   35  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   $CODE$                                 1136  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)




;                                  Library Statistics
;
;                                                       -------- Symbols --------        Pages        Processing
;       File                                            Total     Loaded    Percent      Mapped       Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1                        18619         98          0      1000           00:01.9





;                                            COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:OBJECT/OBJ=OBJ$:OBJECT MSRC$:OBJECT/UPDATE=(ENH$:OBJECT)

; Size:          1136 code + 1407 data bytes
; Run Time:          00:28.9
; Elapsed Time:      01:04.7
; Lines/CPU Min:      2914
; Lexemes/CPU-Min: 29045
; Memory Used:  200 pages
; Compilation Complete
```

0044 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY